UNNPred-CloudHPC Project – Installation Guide for Benchmarks and Programs

Author: Esma Yildirim
Date: Jul 31, 2024

NAS Parallel Benchmark

NAS Parallel Benchmark Installation on AWS:

NPB and NPB-MZ Benchmarks are selected.

1. Download https://www.nas.nasa.gov/assets/npb/NPB3.4.2.tar.gz and https://www.nas.nasa.gov/assets/npb/NPB3.4.2-MZ.tar.gz with wget and untar them

```
$ wget https://www.nas.nasa.gov/assets/npb/NPB3.4.2.tar.gz
$ wget https://www.nas.nasa.gov/assets/npb/NPB3.4.2-MZ.tar.gz
$ tar -xvf NPB3.4.2.tar.gz
$ tar -xvf NPB3.4.2-MZ.tar.gz
```

2. Go to NPB3.4-MPI and NPB3.4-OMP folders under the main directory of NPB3.4.2. Under the config folder copy make and suite templates to original files

```
$ cd NPB3.4.2/ NPB3.4-MPI/config/
$ cp make.def.template make.def.template
$ cp suite.def.template suite.def
```

3. Change the suite.def file and add lines for the applications for different class types then from the main directory of MPB3.4-MPI make suite

```
$ cd ..
$ make suite
```

4. Repeat the same steps for OMP and MZ benchmarks. Remember not all benchmarks are available in all class types

Keras Benchmark Applications

Tensorflow GPU Installation on IC2 cluster at BNL

The new version of tensorflow 2.16.1 has problems finding GPUs. And it is the only version that has Keras 3. The example programs on Keras.io are written mostly on Keras 3.

```
$ module load python/3.11-anaconda-2023.07
$ conda create -n lastenv python=3.9
$ conda activate lastenv
$ module load cuda
$ pip install tensorflow[and-cuda]
```

This will install tensorflow 2.16.1 based on cuda 12.2

SLURM Batch Script example to run Image Classification Resnet with Keras 3:

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -A csihpc
#SBATCH -t 00:30:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --gres=gpu:2
#SBATCH -o hostname_%j.out # File to which STDOUT will be written
#SBATCH -e hostname_%j.err # File to which STDERR will be written

module load python/3.11-anaconda-2023.07
eval "$(conda shell.posix hook)"
conda activate lastenv

export NVIDIA_DIR=$(dirname $(dirname $(python -c "import
nvidia.cudnn;print(nvidia.cudnn.__file__)")))
export LD_LIBRARY_PATH=$(echo ${NVIDIA_DIR}/*/lib/ | sed -r
's/\s+/:/g')${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}

srun time python3 image_classification_from_scratch.py
```

Application 1: Image Classification with ResNET model

Image_classification_from_scratch.py from Keras.io

This application does a binary classification between cat and dog images based on a ResNet model. The original dataset is problematic as some of the images are corrupt. However, the program tries to identify and remove them. But running the program on the IC2 cluster showed additional errors. So, I used another program to identify the corrupt images:

culprit.py

```
from pathlib import Path
import imghdr

data_dir = "./PetImages/Dog/"
image_extensions = [".png", ".jpg"]  # add there all your images file
extensions

img_type_accepted_by_tf = ["bmp", "gif", "jpeg", "png"]
for filepath in Path(data_dir).rglob("*"):
    if filepath.suffix.lower() in image_extensions:
        img_type = imghdr.what(filepath)
        if img_type is None:
            print(f"{filepath} is not an image")
        elif img_type not in img_type_accepted_by_tf:
            print(f"{filepath} is a {img_type}, not accepted by TensorFlow")
```

I removed the identified images in the Cat and Dog folders. The curified dataset is put under `/hpcgpfs01/scratch/eyildirim/PetImages/` folder.

RADICAL PILOT Installation on AWS

radical.pilot and radical.saga must be installed otherwise it creates mpi problems. **psij-python** package must be uninstalled. It runs with Python 3.7 and radical 1.6.0.

```
$python -m venv ve-rct
$ source ve-rct/bin/activate
$pip install radical.pilot
$pip install radical.saga
$pip uninstall psij-python
```

The resource environment is mpirun.

KERAS Application Image Classification on AWS – CPU only

Tensorflow's MultiWorkerMirroredStrategy is designed for clusters with GPUs. To run a multi-threaded version on a single node MirroredStrategy class must be used. Also, cpu-bind must be set to none so that Slurm will not force the threads into a single core.

Installation of Tensorflow with Keras 3:

```
$ python3.9 -m venv ve.keras
$ source ve.keras/bin/activate
$ pip3.9 install tensorflow
$ pip3.9 install radical.pilot
$ pip3.9 install radical.saga
$ pip3.9 uninstall psij-python
$ pip3.9 install urllib3==1.26.15
```

Batch size is reduced to 32 as memory can become a problem

The files are uploaded to github under metric-collection/AWS/

RP Installation and Test Run on Polaris

The base environment has all the necessary packages

```
$ module use /soft/modulefiles ; module load conda; conda activate base
$CONDA_NAME=$(echo ${CONDA_PREFIX} | tr '\/' '\t' | sed -E 's/mconda3|\/base//g' | awk '{print $NF}')
$ VENV_DIR="$(pwd)/venvs/${CONDA_NAME}"
$ mkdir -p "${VENV_DIR}"
$ python -m venv "${VENV_DIR}" --system-site-packages
$ source "${VENV_DIR}/bin/activate"
```

```
$ python3 -m pip install git+https://github.com/radical-cybertools/radical.pilot.git@devel
```

```
$ python3 -m pip install radical.saga
```

Add the following line to pre_exec list for the rp tasks

```
export MPICH_GPU_SUPPORT_ENABLED=0
```